





MMMR 포팅 매뉴얼

1. 개요
2. 개발/실행 환경 정보
2.1 Java 및 JVM
2.2 빌드 도구 및 환경
2.3 웹 애플리케이션 서버 (WAS)
2.4 웹 서버
2.5 Docker 환경
2.6 ROS2 환경
3. 실행 및 배포 절차
3.1 Jenkins 파이프라인 기준 실행 흐름 (CI/CD)
3.2 로컬 실행 시
3.3 로컬 환경 설정
3.4 백엔드 Dockerfile 정보
3.5 프론트엔드 Dockerfile 정보
3.6 docker-compose.yml 정보
3.7 ROS 파일 빌드 및 설정
3.8 Cloudflare 설정 가이드
3.9 자율 주행 및 홈캠 기능 실행 가이드
4. DB 설치 및 초기 데이터 반영
4.1 사용 DB
4.2 초기 설정
5. 주요 테이블 및 매핑 객체
6. 외부 서비스 정보
7. 환경 변수
8. 시연 시나리오
 1차 거울 시연
 모바일 시연
 2차 거울 시연
 마무리

1. 개요

본 문서는 MMMR 프로젝트의 소스코드를 GitLab에서 클론한 이후, 빌드 및 배포를 정상적으로 수행하기 위한 포팅 매뉴얼입니다. 폴더명은 `exec` 입니다.

2. 개발/실행 환경 정보

2.1 Java 및 JVM

- Java Version: **17.0.14**
- JVM: **OpenJDK 64-Bit Server VM (build 17.0.14+7-Ubuntu-122.04.1)**

2.2 빌드 도구 및 환경

- 빌드 도구: **Gradle 8.x 이상**
- 빌드 파일: `build.gradle`
- IDE 권장: IntelliJ IDEA (Ultimate 또는 Community), Eclipse 지원 가능

2.3 웹 애플리케이션 서버 (WAS)

- 내장 WAS: **Apache Tomcat 10.1.31 (Spring Boot 내장)**
- 실행 방식: Spring Boot 내장 Tomcat 실행 (별도 WAS 설치 불필요)

2.4 웹 서버

- Web Server: **Nginx 1.18.0 (Ubuntu)**
- 역할: 정적 리소스 제공, 리버스 프록시 설정 등

2.5 Docker 환경

- Docker Version: **28.0.4, build b8034c0**
- Docker Compose Version: **v2.5.0**

2.6 ROS2 환경

- ROS2 : eloquent (20200124 release)
- Python : 3.7.5
- openssl : 1.0.2u
- choco : 0.10.15
- cmake : 3.27.0 (Visual Studio 2022 Community 를 사용함)
- opencv: 3.4.6
- rti : 5.3.1
- openslice : 6.9.190403
- tensorflow : 1.15

3. 실행 및 배포 절차

3.1 Jenkins 파이프라인 기준 실행 흐름 (CI/CD)

1. GitLab의 **develop** 브랜치에서 소스코드를 클론합니다.
2. **.env** 파일을 **./BE** 디렉토리로 복사한 후, 백엔드 빌드를 진행합니다.
3. 빌드 후 **docker-compose** 명령어를 사용해 컨테이너를 정리 및 재배포합니다.

```
# 소스 클론 (Jenkins Pipeline 내부)
git clone --branch develop https://lab.ssafy.com/s12-mobility-smarthome-sub1/S12P21A703.git

# .env 파일 복사
cp ../../.env ./BE/.env

# 백엔드 빌드
cd BE
chmod 770 ./gradlew
./gradlew clean build -x test

# Docker 재배포
cd ..
docker-compose down
docker-compose build
docker-compose up -d
```

- Jenkins 파이프라인 내에서 각 단계 성공/실패 시 로그 출력이 포함되어 있음
- **.env** 파일이 프로젝트 루트가 아닌 **BE** 하위에 위치해야 하며, **pwd** 및 **cat .env** 등을 통해 환경값 확인 가능함

3.2 로컬 실행 시

```
# GitLab에서 소스 클론
git clone 'https://lab.ssafy.com/s12-mobility-smarthome-sub1/S12P21A703.git'
cd ./BE

# 빌드
chmod 770 ./gradlew
./gradlew clean build
```

```
# 실행
java -jar build/libs/mmmr-0.0.1-SNAPSHOT.jar
```

3.3 로컬 환경 설정

- Java 17 이상 설치 필수
- Gradle 설치 불필요 (`./gradlew` 사용)
- `.env` 파일로 환경 변수 관리 (spring-dotenv 사용)

3.4 백엔드 Dockerfile 정보

```
FROM openjdk:17-alpine
WORKDIR /app
COPY ./build/libs/*SNAPSHOT.jar ./app.jar
EXPOSE 8088
ENTRYPOINT ["sh", "-c", "java -jar app.jar"]
```

- `build/libs` 경로에서 `.jar` 파일을 복사해 컨테이너에 배포합니다.
- 내부 포트는 **8088**를 사용하며, 외부 Nginx 또는 리버스 프록시에서 이 포트를 매핑해 접근합니다.
- 엔트리포인트는 쉘을 통해 `java -jar app.jar` 명령어를 실행합니다.

3.5 프론트엔드 Dockerfile 정보

```
# --- 빌드 단계 ---
FROM node:20-alpine AS builder
WORKDIR /app

COPY . .
RUN yarn install && yarn build

# --- 실행 단계 ---
FROM node:20-alpine AS runner
WORKDIR /app

# 프로덕션 실행에 필요한 최소 파일만 복사
COPY --from=builder /app/public ./public
COPY --from=builder /app/.next ./next
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./package.json

ENV PORT 5173
EXPOSE 5173
CMD ["yarn", "start"]
```

- 2단계 빌드 방식 사용: `builder` 는 Next.js 애플리케이션을 빌드하고, `runner` 는 실제 실행을 담당합니다.
- Next.js 공식 Github의 Dockerfile을 참고했습니다.
 - <https://github.com/vercel/next.js/blob/canary/examples/with-docker/Dockerfile>
- 프로덕션 환경에서 필요한 최소 파일만 복사하여 컨테이너 크기를 줄입니다.
- 내부 포트는 **5173**으로 설정되어 있으며, ENV로 오버라이드 가능함.

3.6 docker-compose.yml 정보

```
version: '3.8'
```

```

services:
  backend:
    container_name: mmmr-backend
    build:
      context: ./BE
      dockerfile: Dockerfile
    env_file:
      - ./BE/.env
    network_mode: host
    expose:
      - "8088"

  frontend:
    container_name: mmmr-frontend
    build:
      context: ./FE/mmmr
      dockerfile: Dockerfile
    network_mode: host
    expose:
      - "5173"
    depends_on:
      - backend

  frontend_mobile:
    container_name: mmmr-frontend-mobile
    build:
      context: ./FE/mmmr_mobile
      dockerfile: Dockerfile
    network_mode: host
    expose:
      - "3000"
    depends_on:
      - backend

```

- 백엔드와 프론트엔드, 모바일 프론트 각각을 컨테이너로 구성하여 `network_mode: host` 로 실행
- 포트는 각각 8088, 5173, 3000번 사용
- 프론트 서비스는 백엔드에 의존성을 설정해 먼저 기동되도록 구성

3.7 ROS 파일 빌드 및 설정

x86 Native Tools Command Prompt for VS 2022(또는 2019)에서 빌드를 진행합니다.

```

# 1. ROS2 설치 위치에서 setup 파일 실행
call C:\dev\ros2_eloquent\setup.bat

# 2. autonomous 폴더 위치로 이동
cd C:\Users\SSAFY\Desktop\S12P21A703\EM\autonomous

# 3. 빌드 실행
colcon build

```

3.8 Cloudflare 설정 가이드

1. 클라우드플레이어 설치
2. 도메인 구매
3. 대시보드에서 도메인 등록 및 cloudflare 네임서버로 변경

4. 클라우드플레어 터널 로그인

나타나는 브라우저에서 원하는 도메인 선택 후 인증

```
./cloudflared tunnel login
```

5. 터널 생성

터널 아이디의 json 파일이 생성됨

```
cloudflared tunnel create <TUNNEL-NAME>
```

6. 구성 파일 생성 (~/.cloudflared/config.yml)

```
tunnel: <TUNNEL-ID>
credentials-file: 위치/터널아이디.json
ingress:
  - hostname: <your-domain.com>
    service: <http://localhost:5000>
  - service: http_status:404
```

7. DNS 자동 연결

```
./cloudflared tunnel route dns <TUNNEL-NAME> <your-domain.com>
```

8. Flask 서버 실행

```
ros2 run sub2 streaming
```

9. 터널 실행

```
./cloudflared.exe tunnel run <TUNNEL-NAME>
```

3.9 자율 주행 및 홈캠 기능 실행 가이드

빌드 완료 후 다음 단계로 실행합니다:

1. SSAFY 브릿지 런치파일 실행

```
cd C:\\Users\\SSAFY\\Desktop\\S12P21A703\\EM\\autonomous
.\\bridgelaunch.bat
```

2. 자율 주행 런치파일 실행

```
cd C:\\Users\\SSAFY\\Desktop\\S12P21A703\\EM\\autonomous
.\\auto_drive.bat
```

3. 스마트 미러 연결 클라이언트 파일 실행

```
cd C:\\Users\\SSAFY\\Desktop\\S12P21A703\\EM\\autonomous
.\\ros2_local_set.bat
ros2 run sub2 nav_command_ws
```

4. 홈캠 영상 스트리밍 파일 실행

```
cd C:\\Users\\SSAFY\\Desktop\\S12P21A703\\EM\\autonomous
.\\ros2_local_set.bat
ros2 run sub2 streaming
```

5. Cloudflare 터널 실행

```
# cloudflare 설치 위치에서 실행
.\cloudflared tunnel run <TUNNEL-NAME>
```

4. DB 설치 및 초기 데이터 반영

4.1 사용 DB

- **MySQL 8.0.41** ([mysql Ver 8.0.41-0ubuntu0.22.04.1 for Linux on x86_64](#))
- JDBC 연결 사용
- 개발환경에서는 **H2 DB** 테스트 가능

4.2 초기 설정

- `application.yml` 또는 `.env` 파일에서 DB 접속 정보 설정
- 초기 테이블 생성은 JPA 자동 생성 사용 가능
- 초기 데이터 삽입: `/resources/data.sql` 또는 Spring Batch 작업으로 처리 가능

5. 주요 테이블 및 매핑 객체

테이블명	엔티티 클래스	레포지토리	서비스	컨트롤러	기타 구성요소
accounts	AccountEntity	AccountRepository	AccountService , AuthService , EmailService	AccountController , AuthController , EmailController	-
profiles	ProfileEntity	ProfileRepository	ProfileService	ProfileController	-
schedules	ScheduleEntity	ScheduleRepository	ScheduleService	ScheduleController	-
todos	TodoEntity	TodoRepository	TodoService	TodoController	-
metros , buses	MetroEntity , BusEntity	MetroRepository , BusRepository	TransportationService	TransportationController	BusClient , MetroClient
metro_informations , businformations	MetroInformationEntity , BusInformationEntity	MetroInformationRepository , BusInformationRepository	-	-	-
home_devices	HomeDeviceEntity	HomeDeviceRepository	HomeDeviceService	HomeDeviceController	-
news	NewsEntity	NewsRepository	NewsService	NewsController	NewsScheduler
- (외부 API 연동)	-	-	-	-	VideoClient , WeatherClient , GeocoderClient

- Redis 캐시 사용 → `application.yml` 내 캐시 설정
- QueryDSL 사용 → `Q클래스` 는 빌드시 자동 생성됨 (`build/generated/querydsl`)

6. 외부 서비스 정보

- Redis 서버 (CLI: **redis-cli 6.0.16**)
- Spring Mail 사용 (SMTP 설정 필요)
- JWT 인증 (jwt 사용)
- Swagger API 문서 제공 (`/swagger-ui/index.html`)
- QueryDSL, Spring Batch, Excel/CSV 처리 기능 포함

7. 환경 변수

```

# Database
DB_URL=jdbc:mysql://localhost:33306/mmmr?useSSL=false&serverTimezone=UTC&characterEncoding=UTF-8&max
DB_USERNAME=root
DB_PASSWORD=mysqlformmmr703
JPA_DDL_AUTO=update

# Redis
REDIS_HOST=j12a703.p.ssafy.io
REDIS_PORT=6379
REDIS_PASSWORD=redisformmmr703
REDIS_TTL=3600000

# Server
SERVER_PORT=8088

# JWT
JWT_SECRET=WatRcfvnLYxwZ9LjxpmwI5ojfgwSqD7G1gaMVkNe7fqv6llm1Qh5eYSUbhZxNig3Z1kkJHKiByYVX8×9Xsa
JWT_ACCESS_TOKEN_EXPIRATION=3600000
JWT_REFRESH_TOKEN_EXPIRATION=604800000
JWT_TEMPORARY_TOKEN_EXPIRATION=3600000

# Logging
LOG_LEVEL_ROOT=INFO
LOG_LEVEL_SPRING_WEB=DEBUG
LOG_LEVEL_HIBERNATE=DEBUG
LOG_LEVEL_APP=DEBUG
LOG_LEVEL_MYBATIS=TRACE

#Frontend Url
FRONTEND_URL=http://localhost:5173

# File Storage Path
FILE_STORAGE_PATH=/var/www/mmmr/uploads

# Batch Configuration
BATCH_JOB_ENABLED=false
BATCH_INITIALIZE_SCHEMA=always

# CSV 파일 저장 경로
CSV_UPLOAD_DIR=${FILE_STORAGE_PATH}/csv

# Mail
MAIL_USERNAME=mmmra703@gmail.com
MAIL_PASSWORD=dwybatnvzsfblqf

# OpenWeatherMap API 설정
OPENWEATHER_API_KEY=af0cbf1d5f5f656dbdf76d2a617f9118
OPENWEATHER_API_URL=https://api.openweathermap.org/data/2.5

# 위도, 경도 API 설정
VWORLD_API_KEY=F7680174-5635-3CCC-B685-DAA021233052
VWORLD_GEOCODER_URL=http://api.vworld.kr/req/address

# Transportation API 설정
METRO_API_KEY=6f63494a466b756e3936754e74686f
METRO_API_URL=http://swopenapi.seoul.go.kr/api/subway
BUS_API_KEY=N30yw4qP23zFDNy7BRoj3kXorHQuiQGwZtRxHnDdOXm%2BPllxkosAsiPz9dw%2FCXxvwfE5T5×0ME

```

```
BUS_API_URL=http://ws.bus.go.kr/api/rest/arrive/getArrInfoByRouteList
```

```
# Youtube api key
```

```
YOUTUBE_API_KEY=AlzaSyCb4rrXLZJBLMrGRKBOKG7LuEs62Km7Cc8
```

```
YOUTUBE_API_URL=https://www.googleapis.com/youtube/v3/search
```

8. 시연 시나리오

✅ 1차 거울 시연

- 음성 제어 시연 (미미야 호출)

- 인삿말, 시간/날씨/타이머/뉴스/일정 확인
- 뉴스는 주제 스캔 후 특정 뉴스 호출

- 홈 카메라 제어

- 거실로 이동, 끄기

- IoT 제어 시뮬레이터 시연

- IoT 목록 확인
- TV/전등 제어

- 무드등 실물 제어 시연

- 무드등 켜기/색상 변경/끄기

- 미디어 제어

- 플레이리스트 영상 재생

📱 모바일 시연

- 사용자 등록

- 프로필 이름·비서명(예: 도윤/해태) 설정

- 정보 입력

- 할 일 추가/완료 처리
- 교통 정보 입력 (버스/지하철, 위치 등)

✅ 2차 거울 시연

- 모바일에서 등록한 정보 활용 시연 (비서명: 해태야)

- 교통정보 확인
- 할일 확인 (완료된 항목 제외)

🎬 마무리

- 시연 종료 안내
-